Location Recommendation in Location-based Social Networks using User Check-in Data

Hao Wang
Dept. of Computer Science
The Univ. of Hong Kong
hwang2@cs.hku.hk

Manolis Terrovitis
Inst. for the Mgmt of Inf. Sys.
Research Center 'Athena'
mter@imis.athena-innovation.gr

Nikos Mamoulis
Dept. of Computer Science
The Univ. of Hong Kong
nikos@cs.hku.hk

ABSTRACT

This paper studies the problem of recommending new venues to users who participate in location-based social networks (LBSNs). As an increasingly larger number of users partake in LBSNs, the recommendation problem in this setting has attracted significant attention in research and in practical applications. The detailed information about past user behavior that is traced by the LBSN differentiates the problem significantly from its traditional settings. The spatial nature in the past user behavior and also the information about the user social interaction with other users, provide a richer background to build a more accurate and expressive recommendation model.

Although there have been extensive studies on recommender systems working with user-item ratings, GPS trajectories, and other types of data, there are very few approaches that exploit the unique properties of the LBSN user check-in data. In this paper, we propose algorithms that create recommendations based on four factors: *a*) past user behavior (visited places), *b*) the location of each venue, *c*) the social relationships among the users, and *d*) the similarity between users. The proposed algorithms outperform traditional recommendation algorithms and other approaches that try to exploit LBSN information.

To design our recommendation algorithms we study the properties of two real LBSNs, Brightkite and Gowalla, and analyze the relation between users and visited locations. An experimental evaluation using data from these LBSNs shows that the exploitation of the additional geographical and social information allows our proposed techniques to outperform the current state of the art.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: [Miscellaneous]

General Terms

Application, Experimentation

Keywords

location-based social network, recommender system, personalized PageRank, bookmark-coloring algorithm, check-in data, geo-filtering

1. INTRODUCTION

In location-based social networks (LBSNs), users share information about their locations, the places they visit and their movement alongside with other social information. Visits are reported explicitly (by user *check-ins* in known venues and locations) or implicitly by allowing smartphone applications to report visited locations to the LBSN. This information is then shared with other users who are socially related (e.g., friends). The same information can be exploited by the LBSN operator to propose new points of interest to users. Recommending new locations is an important issue; it allows to efficiently advertise companies with a physical presence (theaters, bars, restaurants, etc.) and create revenue for the LBSN operator.

Recommender systems are widely used and they have been studied in research quite extensively. The most popular approach in recommender systems is that of *collaborative filtering*, where recommendations are created based on whether a user has purchased a product in the past and on whether she liked it or not. Using the past behavior of a user, new recommendations are created based on the similarity of users or the similarity of products (items). While these algorithms can be adjusted to the problem of recommending new locations to users, by taking into account previous user check-ins, significant information like the distance of the proposed location to the user neighborhood or the social interaction between the user and those users that have visited this location are ignored.

By studying real LBSN data we can confirm that the social links of a user and the distance of a location to her previous visits are important factors in predicting whether she will visit a new location or not. By analyzing the publicly available data of Gowalla [4] we show that more than 80% of the new places visited by a user are in the 10km vicinity of previous check-ins and more than 30% of the new places visited by a user have been visited by a friend or a friend-of-a-friend in the past. These facts imply that geographical and social information significantly affect the choices of a user when deciding which new place to visit.

While there are several papers that study the behavior of users in LBSNs, works that exploit geographical and social information for creating recommendations have only recently appeared. The most closely related work to ours is the work of [16], which proposes a recommendation algorithm that takes into account the rich knowledge of the LBSNs. It is shown that the additional information allows for more accurate recommendations than those created by traditional algorithms. Our proposal delves deeper in the properties of LBSNs, that people often visit new locations that are geographically close to their past visited locations, and such new visits are usually influenced by their social relationships. Based on such properties, we propose recommendation algorithms that outperform the methods of [16] by a wide margin. The contribution of the paper can be summarized as follows:

- We analyze datasets from real LBSNs (Brightkite and Gowalla) and we identify several useful correlations between visited venues, their location and social information about the users of the network.
- We propose new recommendation algorithms that exploit the correlations in the data and perform superiorly to the current state of the art.
- We experimentally evaluate the proposed techniques in real work data.

2. DATASETS AND DATA ANALYSIS

In this section, we focus on the analysis of two LBSN datasets, Brightkite and Gowalla [4], both of which are publicly available from the Stanford network analysis project (SNAP)[1]. We present the basic statistical properties and we then focus on the importance of the geographical and social factors in the choice of the venues that are visited by a user. In brief, our findings are that *a*) most visits in the data are first-time visits, *b*) people usually visit nearby locations; and *c*) friendships are related to the choice of venues that are visited by users.

2.1 Datasets

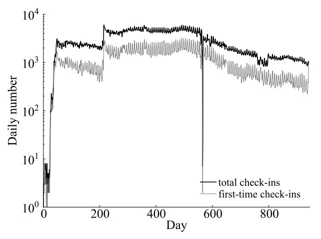
Brightkite was an LBSN created in 2007 and closed after April 2012. The dataset contains 2,627,870 check-in records made by 58,228 users involving 772,933 locations over 942 days from 21 Mar 2008 to 18 Oct 2010. Among all the users there are in total 214,078 friendship links.

Gowalla was an LBSN launched in 2007, purchased by Facebook in 2011, and closed in 2012. The dataset contains 6,264,203 check-in records made by 196,591 users involving 1,280,956 locations over a time period of 627 days from 04 Feb 2009 to 23 Oct 2010. The dataset also contains 950,327 friendship links among users.

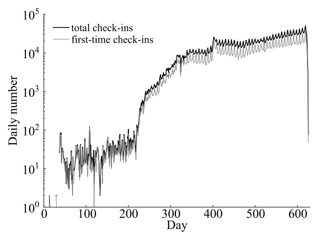
2.2 First-time Visits

Our main interest in the data lies in *first-time visits*, i.e., the event that a user visits a venue for the first time. We believe that recommending to a user a *new* location, where she has never been before, is of great importance, while recommending some already visited location is not as useful. Still, every past visit is of importance to algorithms that make recommendations.

[1]http://snap.stanford.edu/index.html



(a) Brightkite



(b) Gowalla

Figure 1: Daily number of check-in records in Brightkite and Gowalla.

The majority of visits in the available data are first-time visits. This does not necessarily reflect the fact that users constantly visit new places; it is probable that users are more enthusiastic when they first visit a place, but omit to report recurring visits to a place. The daily numbers of check-ins and the daily numbers of first-time visits for both Brightkite and Gowalla datasets are shown in Figure 1. From the figures we see that Gowalla shows a distinct trend of growing, whereas Brightkite is shrinking after around the 500th day. In both cases among all the check-in records (black bold lines) a certain portion are first-time visits (gray lines). In fact, in Brightkite 40.83% of the check-ins are first-time visits, and in Gowalla this number goes up to 63.56%.

We are also interested in establishing whether the large number of first-time visits is attributed to a few particular very active users (for instance, world travellers) or whether it is a common behavior of many users. Figure 2 presents a more detailed view of the first-time visits in the data. For each user u, we take her check-in records C_u and the set of locations L_u that she has ever visited. The quantity $r_{new} = |L_u| / |C_u|$ thus describes how often user u visits new locations. Figure 2 depicts the (cumulative) distribution of r_{new} in both datasets. It shows how many users have a larger ratio of new visits r'_{new} for every value of r_{new} from 0 to 100%. A point on the horizontal axis represents a value of r_{new}, and the corresponding point on the vertical axis shows the ratio of users of which have $r'_{new} \geq r_{new}$.

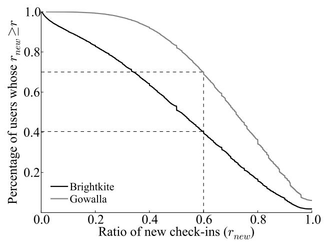


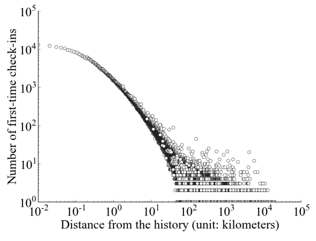
Figure 2: Cumulative distribution of r_u.

To prevent bias from less active users who only visit a small number of locations and thus have an r_{new} very close to 100%, we consider only those users with sufficient many check-in records ($|C_u| \geq 10$). It turns out that, as shown in Figure 2, reporting visits mostly on new locations is indeed a common behavior of the majority of the users. Even if recurring visits are omitted, the data shows that users visit many new locations and that they are enthusiastic enough to report them. This fact strengthens the motivation for creating a recommendation algorithm for unvisited venues.

2.3 Importance of Nearby Locations

Although in both datasets the locations are distributed all over the world, from real-life experience we know that people do not frequently visit locations far away from their usual active area (e.g., home, office, etc.). To verify this hypothesis, we have calculated the following statistics. For each user u, we take the list of locations that she has visited, $L_u = (\ell_{u,1}, \ell_{u,2}, \cdots, \ell_{u,n})$, sorted in order of their first time of visit. Then, for each newly visited location $\ell_{u,i}$ ($i > 1$), we calculate the geographical distance between $\ell_{u,i}$ and the history $H_{u,i} = \{\ell_{u,1}, \ell_{u,2}, \cdots, \ell_{u,i-1}\}$, which is defined as $\text{geodist}(\ell_{u,i}, H_{u,i}) = \min_{j<i} \text{geodist}(\ell_{u,i}, \ell_{u,j})$, where $\text{geodist}(\ell_1, \ell_2)$ is the geographical distance between the two locations ℓ_1 and ℓ_2. This distance measures how far a newly visited location is from the closest location in the user's past visits list. Intuitively, it measures how far are the new places that are visited by a user, from the places she has visited in the past.

Figure 3 shows the distributions of such distance values in both datasets. The horizontal axis traces the distance value



(a) Brightkite (b) Gowalla

Figure 3: Distribution of the distance between a newly visited location and previously visited locations.

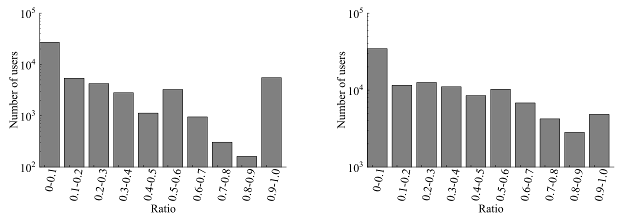
d, and the vertical axis the total number of user-location pairs (u, ℓ) such that, when u visited ℓ for the first time, the distance between ℓ and u's previous visited locations is d. Note that both axes are in log scale.

While Brightkite and Gowalla differ in percentage of first-time visits, they exhibit highly similar distributions of check-in distance values. Most check-ins happen near some past check-ins. In particular, a small radius of 10 kilometers, covers 67.57% of first-time visits in Brightkite and 81.93% in Gowalla.

This observation shows that taking proximity into account when recommending a new venue to a user can increase the quality of the recommendation.

2.4 Importance of Friendships

The impact of social interaction in the choice of venues has already been identified in research literature (and used for recommendations [23, 24]), and it is something we expect from real world experiences. To assess how friends and the broader community of a user affect her choice of places, we measure the following statistics: for each user u who has visited venues $L_u = \{\ell_{u,1}, \ell_{u,2}, \cdots, \ell_{u,n}\}$, we calculate the percentage of places in L_u that have been visited by a friend of u or a friend-of-friend of u prior to u's first visit to them. The results for both datasets are depicted in Figure 4. The horizontal axis in Figure 4 traces the percentage of first-time visits that take place in a venue that has been previously visited by a friend or a friend-of-friend and the vertical axis shows the number of users whose behavior matches this percentage.



(a) Brightkite (b) Gowalla

Figure 4: Distribution of users according the impact of the community in the places they choose to visit.

The results of the aforementioned experiment show that around 31% of the first time visits in Gowalla and 23% in Brightkite take place in a venue that has been previously visited by a friend or a friend-of-friend. Given that the average size of the friends-of-friends community is around 1% of total users in Gowalla and 0.5% in Brightkite, we believe that the community of friends is an important factor that affects the users when choosing a new place to visit.

3. RECOMMENDATION MODEL AND BACKGROUND

This section formalizes the recommendation problem and presents the necessary background for describing our algorithms in Section 4. We complement the section with the brief presentation of recommendation techniques which will be used as a point of reference for our contribution. These techniques include current state-of-the-art methods and naïve methods which model the impact of individual factors in the user's choice of venues.

3.1 Problem Formalization

The original data is a series of records that describe users $\mathcal{U}$, locations $\mathcal{L}$ and check-ins $\mathcal{C}$. $\mathcal{C}$ is a set of pairs (v, t), where v is a visit that associates a user with a place, and t is the timestamp of the visit. We represent an LBSN as a graph $G = (U, L, E_F, E_V, C_V)$. U are nodes representing users from $\mathcal{U}$; L are nodes representing locations from $\mathcal{L}$. $E_F \subseteq U \times U$ are edges, which represent the friendship links between users; $E_V \subseteq U \times L$ are edges which represent the visits of users to locations; and C_V are weights attributed to each edge from E_V, counting the number of visits performed by a single user $u \in U$ to a single location $\ell \in L$. We consider all edges undirected. The aforementioned model is broad enough to support every recommendation algorithm we present in this work. Still, not all information is exploited by every technique. An example of a graph G is depicted in Figure 5. User nodes (u_1–u_3) are represented by circles and location nodes (ℓ_1–ℓ_3) are represented by triangles. The figure includes two friendship edges (f_1, f_2) and four visit edges (v_1–v_4) with their weights. In the rest of the paper, we omit the labels for the edges and show only the weights in the visit edges to provide less verbose examples.

The recommendation problem that we are studying is formalized as follows. Given a network G and a user u, create a list of N location recommendations, that have not yet been visited by user u. Recommendations are successful if the locations are visited by the user in a specified time window

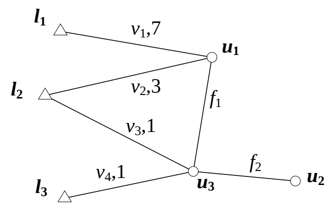


Figure 5: An example of a social network graph

in the future, e.g., two months after a recommendation is made.

3.2 Background

The proposed algorithms reuse the idea of personalized PageRank (PPR) [8], which was introduced to endow webpages with a universal ranking. In this section we describe the idea of PageRank and personalized PageRank and we present the bookmark-coloring algorithm we employ to calculate the latter.

The fundamental idea underlying PageRank is a recursive illustration of "important pages" that *important pages are referenced by many important pages*. In a graph G of n nodes, the PageRank value π_i of node $i \in G$ is defined as

$$\pi_i = \frac{1-\alpha}{n} + \alpha\left(\frac{\pi_1}{d_1} + \cdots + \frac{\pi_{i-1}}{d_{i-1}} + \frac{\pi_{i+1}}{d_{i+1}} + \cdots + \frac{\pi_n}{d_n}\right),$$

where α is a constant usually set to 0.85 and d_i is the out-degree of node i. The above definition leads to the following vector form:

$$\boldsymbol{\pi}^T = \alpha \cdot \boldsymbol{\pi}^T \boldsymbol{P} + (1-\alpha) \cdot \frac{1}{n} \cdot \mathbf{1}^T, \qquad (1)$$

where $\boldsymbol{P}$ is the row-normalized adjacency matrix of the graph. Eq. (1) models the Markovian behavior of a *random surfer* in the graph who, at any node, follows a random out-link to a neighbor with probability α, or "teleports" to an arbitrary node with probability $1-\alpha$.

Personalized PageRank (PPR) [8] extends the basic idea of PageRank by replacing the uniformly random teleportation term $\frac{1}{n} \cdot \mathbf{1}^T$ in Eq. (1) by a personalized vector $\boldsymbol{u}_i^T$, in which only the i-th element is 1 and all the others are 0. Given a source node i, the PPR for the whole graph, $\boldsymbol{\pi}$, is given by Eq. (2):

$$\boldsymbol{\pi}^T = \alpha \cdot \boldsymbol{\pi}^T \boldsymbol{P} + (1-\alpha) \cdot \boldsymbol{u}_i^T. \qquad (2)$$

In terms of random surfing, instead of teleporting to a random node in the graph, at any step, the surfer returns to node i with a certain probability, thus this model is also known as *random walk with restart (RWR)*. An element π_j in the steady-state solution of Eq. (2) actually reflects how close node j is to node i.

To efficiently calculate the PPR values of a graph G according to Eq. (2), we use the *bookmark-coloring algorithm* (BCA) introduced in [3]. Algorithm 1 summarizes the basic steps of BCA. Intuitively, BCA assumes that we have a fixed amount of paint on node u (called *bookmark*, Line 3) and emulates the spread of paint to nearby nodes. Each node keeps $1-\alpha$ portion of the paint that it receives and distributes the remaining paint to its neighbors. The transition probabilities are given by Eq. (2) and the process terminates if the paint to be redistributed at each node does not exceed a small constant ϵ.

Algorithm 1 Bookmark-coloring for Computing PPR

```
    BCA($G, u, \alpha, \epsilon$)
    // Input: $G = (U, E)$ is a graph; $u \in U$ is a node; $\alpha$ is
              the constant as in Eq. (2); $\epsilon$ is a tolerance
              threshold
    // Output: PPR vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_{|U|})$
 1: for each $i \in U$ do
 2:     $\pi_i := 0$; $d_i :=$ number of friends of user $i$
 3: Initialize a zero-valued vector $\boldsymbol{b}$ of size $|U|$; set $b_u := 1$
 4: repeat
 5:     for each $i \in U$ do
 6:         if $b_i < \epsilon$ then continue
 7:         $\pi_i := \pi_i + (1-\alpha) \cdot b_i$
 8:         for each friend $j$ of $i$ do
 9:             $b_j := b_j + \alpha \cdot b_i / d_i$
10: until no change in $\boldsymbol{b}$
11: return $\boldsymbol{\pi}$
```

3.3 Baseline Strategies for Location Recommendation

3.3.1 User-based Collaborative Filtering

User-based collaborative filtering (`UserCF`) is based on the idea that *similar users have similar preferences on locations.* To estimate the interest of u in location ℓ, `UserCF` considers $u_1, u_2, \cdots, u_{|U|}$ who have visited ℓ and see how similar they are to u. Intuitively if u is similar to most of these users, then it is highly possible that u will be interested in location ℓ too.

This idea is easily applicable in our setting. For each user u we consider her *visiting profile*, and ignore any friendship information (i.e., E_F edges in G are ignored). Based on the locations that u has visited and their visiting frequencies (v edges on the graph), we create the profile of user u as a vector $\boldsymbol{p}_u$ of length $|L|$,

$$\boldsymbol{p}_u = \left(w_{u,\ell_1}, w_{u,\ell_2}, \cdots, w_{u,\ell_{|L|}}\right), \qquad (3)$$

where w_{u,ℓ_i} is 0 if there is no edge between u and ℓ_i and else $w_{u,\ell_i} \in W_V$ is defined as the normalized frequency of user u visiting location $\ell_i \in L$, i.e., $w_{u,\ell_i} = \text{freq}(u,\ell_i)/\sum_j \text{freq}(u,\ell_j)$.[2]

Cosine similarity can be used to measure the similarity between the visiting profiles of two users. Therefore, the preference of user u on location ℓ can be estimated as

$$\text{score}(u,\ell) = \frac{\sum_{u' \in U} \cos(\boldsymbol{p}_u, \boldsymbol{p}_{u'}) \cdot w_{u',\ell}}{\sum_{u' \in U} \cos(\boldsymbol{p}_u, \boldsymbol{p}_{u'})}. \qquad (4)$$

After estimating u's preference on all the unvisited locations, a recommendation list can be naturally generated by selecting the N locations with the highest scores [1].

3.3.2 Location-based Collaborative Filtering

Location-based collaborative filtering (`LocCF`) comes from the assumption that *people visit similar locations* and it is

[2]More advanced weighting schemes are possible. For example, TF-IDF methods can be used to penalize commonly visited locations (see, for example [13]). However via experiments we found that such advanced schemes were not as good as the simple method here. The reason is that the data is sparse, thus even the most popular location cannot be really considered as "common" in the TF-IDF sense.

a direct application of item-based collaborative filtering [1]. To estimate the preference of user u on location ℓ, instead of comparing the user vectors as in `UserCF`, we compare location profiles $\boldsymbol{p}_\ell$:

$$\boldsymbol{p}_\ell = \left(w_{u_1,\ell}, w_{u_2,\ell}, \cdots, w_{u_{|U|},\ell}\right), \tag{5}$$

where $w_{u_i,\ell}$ is 0 if there is no edge between u_i and ℓ, else $w_{u_i,\ell} = \text{freq}(i,\ell)/\sum_j \text{freq}(j,\ell)$ and

$$\text{score}(u,\ell) = \frac{\sum_{\ell' \in L} \cos\left(\boldsymbol{p}_\ell, \boldsymbol{p}_{\ell'}\right) \cdot w_{u',\ell}}{\sum_{\ell' \in L} \cos\left(\boldsymbol{p}_\ell, \boldsymbol{p}_{\ell'}\right)}. \tag{6}$$

As in `UserCF`, the N new locations with the highest scores are extracted as the final recommendation. Again the friendship information of G is ignored.

3.3.3 Location Nearest Neighbor

Motivated by the observation in Section 2.3 that users tend to visit places nearby their previous whereabouts, we create a simple reference recommender termed *location nearest neighbor* (`LocNN`) based only on the spatial distance of a location to the locations that have been previously visited by a user. Specifically, based on u's visiting history $L_u = \{\ell_1, \ell_2, \cdots, \ell_n\} \subseteq L$, score$(u,\ell)$ of every $\ell \notin L_u$ is simply defined over the geographical distance between ℓ and L_u:

$$\text{score}(u,\ell) = \frac{1}{\min_{\ell' \in L_u} \text{geodist}(\ell,\ell')}. \tag{7}$$

The N unvisited locations with the highest scores are provided as recommendations to u.

3.3.4 Friend-based Collaborative Filtering

Unlike previous methods which do not consider any social relationship, friend-based collaborative filtering (`FriendCF`) proposed in [23] exploits the influence of friendships. `FriendCF` is based on the assumption that *people listen to their friends and follow their friends' recommendations.* `FriendCF` calculates the preference of a target user u on some location ℓ in a similar way to Eq. (4). The only difference is that `FriendCF` considers only u's direct friends F_u, i.e., the users that are directly associated to u with an friendship edge f in G instead of all users U. Thus the score in this case is given by the following equation:

$$\text{score}(u,\ell) = \frac{\sum_{u' \in F_u} \cos\left(\boldsymbol{p}_u, \boldsymbol{p}_{u'}\right) \cdot w_{u',\ell}}{\sum_{u' \in F_u} \cos\left(\boldsymbol{p}_u, \boldsymbol{p}_{u'}\right)}. \tag{8}$$

The N unvisited locations with the highest scores are recommended to user u. `FriendCF` uses all information in G, but in a rather simple way; direct friendship information is only used as a filter in the comparison with similar users.

3.3.5 Random Walk with Restart

The authors of [16] argue that *not only direct friends but also distant friends have influence on user check-in behaviors.* Therefore they employ a random walk with restart model for location recommendation, which we call `RWR`.

Given an LBSN $G = (U, L, E_F, E_V, W_V)$, `RWR` takes a unified view of user nodes and location nodes and also a unified view of friendship and visit edges, thus G is simply a network with $|U| + |L|$ nodes and $|E_F| + |E_V|$ edges between them. To estimate the preference of user u on location ℓ, `RWR` calculates the PPR value of ℓ with respect to u. Specifically, the out-going transition probability from a user $u \in U$ to any node j that is directly associated with u is defined as

$$p_{uj} = \begin{cases} \frac{1}{2|F_u|}, & \text{if user } j \in U \text{ is a friend of } u, \\ \frac{w_{u,j}}{2}, & \text{if location } j \in L \text{ is visited by } u. \end{cases} \tag{9}$$

Here $|F_u|$ is the number of friends of user u, and $w_{u,j}$ is the weight associated with the visiting edge that links location j to u. We assume $w_{u,j} = 0$ if no such edge exists. Since there is no link between locations, the transition probability out of a location $\ell \in L$ is simply $p_{\ell,j} = w_{\ell,j}$, where $j \in U$ is a user and $w_{\ell,j}$ is the weight of the edge between ℓ and j. The score is given by Eq. (2). The N unvisited locations with the highest score are recommended to user u.

`LocNN` and `FriendCF` are aimed to demonstrate the impact of locality and social interaction in user choices, whereas `UserCF` and `LocCF` represent the state of the art in traditional recommendation settings. The `RWR` method is the most closely related work to our own and the most effective recommendation technique that exists in the literature for location recommendation in LBSNs.

4. RECOMMENDATION ALGORITHM

In this section we propose two algorithms for making recommendations in LBSNs. The proposed algorithms are based on the observations we found in Section 2 and they rely on PPR, which we calculate using BCA [3] (Section 3.2).

We believe that, similar to the spread of the paint on the graph, the fame and recommendations spread in the real world social network. Assume that Jane asks one of her friends, Stella, for a new interesting bar. Stella recommends to Jane the places she likes and frequently visits, but she does not stop there; she also reports bars that have been recommended to her by her own friends. These places include of course places that have been recommended by her friends' friends and so on. Finally, Jane decides to visit those places that are highly recommended by her community, even if she does not personally know every person ("everyone goes there!") and that are not far away from her home. In the rest of the section, we propose two algorithms that follow this rationale, but calculate the flow of information and the construct the local community in a different way.

4.1 Friendship-based Bookmark-coloring Algorithm (FBCA)

The idea of the first recommendation algorithm, the *friendship-based bookmark-coloring algorithm* (`FBCA`), is to perform a BCA on G, by taking into account only the friendship edges E_F. We start from the user u (who will be the recipient of the recommendations) and color all users. After all users are colored, we attribute color to the places they have visited and we report the places with the highest color that are within a distance threshold from the visit history of u. The pseudo-code for the algorithm is depicted in Algorithm 2.

For each user, the PPR value is computed (Line 2) and then distributed to her visited locations (Lines 4-6). After filtering out those locations far away from u's visiting history (Line 8), N locations that accumulate the highest PPR values are eventually recommended to the user (Lines 9-13). To carry out the PPR computation at Line 2, we employ BCA (Algorithm 1).

Algorithm 2 FBCA for Location Recommendation

```
   FBCA $(G, u, d, N)$
   // Input: $G = (U, L, E_F, E_V, W_V)$ is an LBSN; $u \in U$
            is the target user; $d$ is a threshold for
            geographical distance; $N$ is the number of
            new locations to recommend
   // Output: A set of $N$ locations, $R$, implemented as a
             priority queue
 1: Get the set of locations that user $u$ has visited, $L_u$
 2: Compute PPR values $\pi_v$ for all $v \in U \backslash \{u\}$ based on the
    user-friendship graph $(U, E_F)$
 3: Initialize $s_\ell := 0$ for each $\ell \in L \backslash L_u$
 4: for each $v \in U \backslash \{u\}$ do
 5:     for each $\ell \in L$ visited by $v$ do
 6:         $s_\ell := s_\ell + \pi_v \cdot w_{v,\ell}$
 7: for each $\ell \in L \backslash L_u$ do
 8:     if geodist$(\ell, L_u) > d$ then continue
 9:     if $|R| < N$ then $R := R \cup \{(\ell, s_\ell)\}$
10:     else if $s_\ell \leq \min(R)$ then continue
11:     else
12:         $R$.pop_min()
13:         $R := R \cup \{(\ell, s_\ell)\}$
14: return $R$
```

4.2 Location-friendship Bookmark-coloring Algorithm (LFBCA)

We assume that information flows through friendship and social interaction links are intuitive and lead to trustworthy recommendations; this is also indicated by the experimental results for FBCA in Section 5. Still, some people are friends online only because they are friends offline or they have other common interests not necessarily related to visiting locations. The importance of using users that behave similarly to the target user u to create recommendation, should not be completely ignored. The *location-friendship bookmark-coloring algorithm* (LFBCA) we propose in this section aims at reconciling social interaction and similarity in a common recommendation algorithm.

To capture both the friendship and similarity relations between users in G, we *augment* $G = (U, L, E_F, E_V, W_V)$ to $\tilde{G} = (U, L, E_S, E_V, W_V, W_S)$, where friendship edges have been replaced with similarity edges and additional similarity edges have been introduced. Moreover, each similarity edge is annotated with a weight $w \in W_S$, which quantifies how similar two users are.

The augmentation procedure is depicted in Figure 6. Figure 6(a) shows the original graph G with friendship edges linking different users. Check-in edges are annotated with their frequencies. The first step is to add a similarity edge between every two users who have visited a common location, i.e., we add the edges $E_S = \{(u, u') \mid \exists \ell \in L \text{ s.t. } (u, \ell) \in E_V \wedge (u', \ell) \in E_V\}$. This leads to Figure 6(b) where additional similarity edges have been added to the graph as dashed edges (we omit locations to simplify the graph). Note that two users can be associated with both a friendship and a similarity edge. Each similarity edge is associated with a tentative weight which is given by the following similarity function:

$$\text{sim}(u_1, u_2) = \cos(\boldsymbol{p}_{u_1}, \boldsymbol{p}_{u_2}), \tag{10}$$

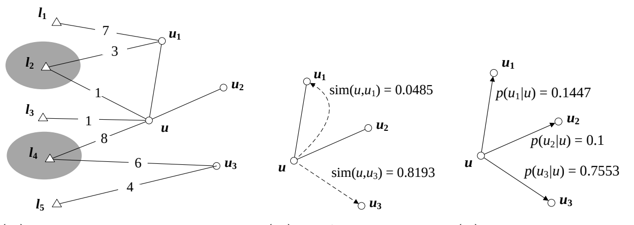
where $\boldsymbol{p}_u$ is given by Eq. (3). In the example of Figure 6(b) we have:

$$\text{sim}(u, u_1) = \cos(\boldsymbol{p}_u, \boldsymbol{p}_{u_1}) = 0.0485,$$
$$\text{sim}(u, u_3) = \cos(\boldsymbol{p}_u, \boldsymbol{p}_{u_3}) = 0.8193.$$

The next step is to merge the friendship edges to the similarity edges. We use a parameter, $\beta \in [0, 1]$, to tune the importance of the similarity and friendship edges. More specifically, regardless of the teleportation, at the node of user u we follow the similarity location-friend links with probability β and follow the original friendship links with probability $1-\beta$. Let F_u and LF_u be the sets of all u's friends and location-friends respectively (those that are associated directly with a similarity edge), and let $S_u = \sum_{v \in LF_u} \text{sim}(u, v)$. The final out-going transition probability at u is now adjusted as

$$p_{u,v} = \begin{cases} (1-\beta) \cdot \frac{1}{|F_u|}, & \text{if } v \in F_u \backslash LF_u, \\ \left(\frac{1-\beta}{|F_u|} + \frac{\beta}{S_u} \cdot \text{sim}(u, v)\right), & \text{if } v \in F_u \cap LF_u, \\ \frac{\beta}{S_u} \cdot \text{sim}(u, v), & \text{if } v \in LF_u \backslash F_u. \end{cases} \tag{11}$$

Note that when $\beta = 0$, $\tilde{G}$ degenerates to the original G we used in FBCA. Using Eq. (11) we update the weights of all similarity edges and we merge friendship and similarity edges, to a single similarity edge in the cases where users were linked with both of them.



(a) The original snapshot (b) Augmented location-friend links (c) Final transition probability

Figure 6: An example of network augmentation.

By applying Eq. (11) to the example in Figure 6 with $\beta = 0.8$, we get the final transition probabilities shown in Figure 6(c). Although not even a friend of u, u_3 gets the highest probability since, judging from their common preference on location ℓ_4, u and u_3 are more like-minded. Also, although both u_1 and u_2 are friends of u, u_1 gets a slight advantage over u_2 because u_1 had some common interest with u on location ℓ_2.

Algorithm 3 summarizes the above method. Note that unlike G in which all edges between users are unweighed,

Algorithm 3 LFBCA for Location Recommendation

```
   LFBCA$(G, u, d, \beta, N)$
   // Input: $G = (U, L, E_F, E_V, W_V)$ is an LBSN; $u \in U$
            is the target user; $d$ is a threshold for
            geographical distance; $\beta$ is the augmentation
            parameter; $N$ is the number of new locations
            to recommend
   // Output: A set of $N$ locations, $R$
 1: Get the set of locations that user $u$ has visited, $L_u$
 2: Augment snapshot $G_t$ into $\tilde{G}_t$ with parameter $\beta$
 3: In $\tilde{G}_t$, compute PPR values $\pi_v$ for all $v \in U \backslash \{u\}$
 4: // Same as Lines 3-13 in Algorithm 2.
 5: return $R$
```

the edges between users in the augmented graph $\tilde{G}$ are all weighted according to Eq. (11). As a result, when computing PPR, Line 9 of Algorithm 1 should be adjusted to handle non-uniform distributions (the adjustment is trivial).

5. EXPERIMENTS

In this section we experimentally evaluate the proposed algorithms `FBCA` and `LFBCA`. As a point of reference we also present results for all other algorithms of Section 3.3 (`UserCF`, `LocCF`, `FriendCF`, `LocNN` , and `RWR`). All algorithms have been implemented in `C++`, and tested in Linux.

5.1 Methodology

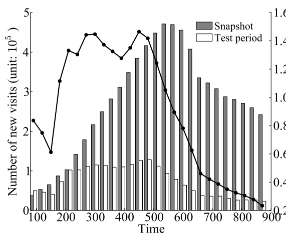
We have tested the algorithms using the Gowalla and Brightkite datasets. In each dataset we defined *snapshots*, which are basically subsets of the complete dataset which describe the condition of the dataset in the past. We used the snapshots to create recommendations, and then we checked whether a user had followed the recommendations during the testing period, i.e., a fixed time period starting at the moment of the snapshot. More specifically, we define a snapshot of a social network G at timestamp t as $G_t = \left(U, L, E_F, E_V^t, W_C^t\right)$, where U, L, E_F are as in G, and E_V^t and W_C^t refer only to the check-ins that have taken place up to time t, i.e., they are built over $\mathcal{C}_t = \{(c, t') \in \mathcal{C} \mid t' < t\}$.[3] A testing period of size *pd* defines a time interval $[t, t + pd)$, during which we check whether a user visited a recommended location or not. Since Brightkite contains check-in records of 942 days and Gowalla 627 days, we split the datasets into the snapshot G_t and the testing data $\mathcal{C}_t^{t+pd}$ based on a sliding timestamp t. We always use a constant time period *pd*. Specifically, we run the recommenders at the end of every month, and evaluate the quality of recommendation using known facts within the next two-month period. In other words, we recommend based on the snapshot G_t, and evaluate the recommendations using the known facts in $\mathcal{C}_t^{t+60} = \mathcal{C}_{t+60} \backslash \mathcal{C}_t$. To guarantee sufficient data for the recommendation and evaluation, we set $t = 90, 120, \cdots, 870$ for Brightkite and $t = 90, 120, \cdots, 510$ for Gowalla. In addition, in all the experiments we consider only *active users*, i.e., users who have at least one new visit in the testing period C_t^{t+60}. Note that in order for all the random walk methods to work, in the snapshots every friendship link is replaced by two directed links.

Figure 7 shows how the number of active users and the number of visits in the snapshot and the testing period evolve over time. Gray bars show the total number of new visits in the snapshot G_t, and white bars in the testing periods $\mathcal{C}_t^{t+pd}$. The curves show the number of active users in the testing periods. The basic conclusion of Figure 7 is that while Gowalla was a growing network, Brightkite was being abandoned by its users after some point and less and less users were active. This observation allows interpreting the rest of the results on the quality of recommendations.

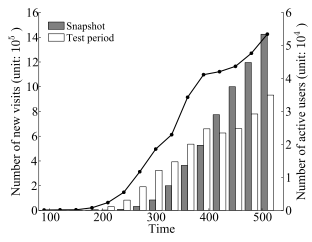
5.2 Metrics

Assume that we have K active users in a testing period $\mathcal{C}_t^{t+pd}$, which contains $\left|\mathcal{C}_t^{t+pd}\right| = T$ testing records, and an

[3]Note that we define the snapshot only in terms of visits; we do not have any information about how the other characteristics of the network evolve (e.g., friendship links, addition on new users, etc.)



(a) Brightkite



(b) Gowalla

Figure 7: Sizes of the snapshots / testing datasets and number of active users

algorithm $\mathcal{A}$ that generates a set of N locations for each active user u, $\mathcal{R}_u = \{\ell_{u,j}\}_{j=1,2,\cdots,N}$. We define the *hits* H_u of the algorithm as

$$H_u = \left|\left\{(u, \ell) \in C_t^{t+pd} \mid \ell \in \mathcal{R}_u\right\}\right|,$$

i.e., the number of recommended locations that are actually visited by the user in the testing period. This number can be interpreted in different ways, each showing one aspect of the recommender:

$$\text{Precision@N}\,(\mathcal{A}) = \sum\nolimits_u H_u/(K \times N),$$
$$\text{Recall@N}\,(\mathcal{A}) = \sum\nolimits_u H_u/T,$$
$$\text{Utility@N}\,(\mathcal{A}) = |\{u \mid H_u > 0\}|\,.$$

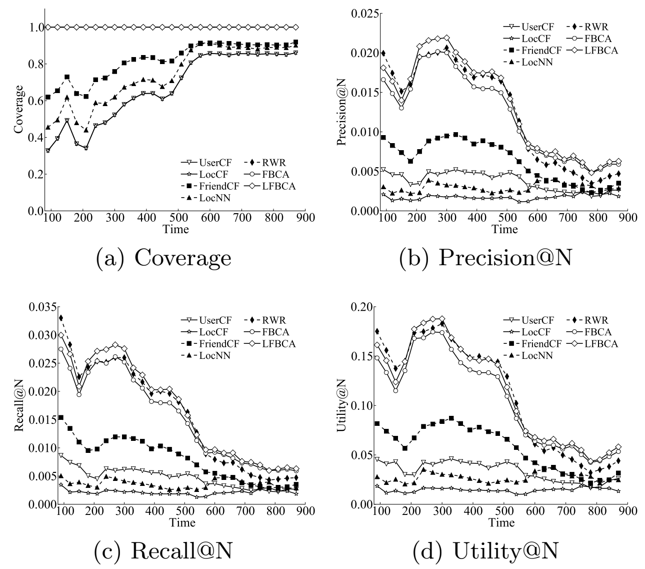
Precision@N and Recall@N are well known metrics in information retrieval and they are used as defined here in the evaluation of the main competitor method, the `RWR`, in [16]. Furthermore, we define the Utility@N metric to capture the quality of the recommendations in terms of the size of the community that follows at least one of them. We perform the evaluation of the proposed methods based on these metrics and we also report the Coverage of each method. The Coverage is the percent of active users for whom the algorithm is able to produce a recommendation.

5.3 Results

The default parameters for the the `FBCA` and the `LFBCA` are $\beta = 0.8$, $d = 2.0$, the snapshot of Gowalla being on the 270th day and for Brightkite on the 600th day. We create $N = 10$ recommendations for each algorithm. Figures 8 and 9 show the results for all algorithms as we examine the snapshots of the LBSNs at the end of each month starting from the 3rd month. The horizontal axis in all graphs depicts the time in days, taking values $t = 90, 120, \cdots, 870$. A first observation is that algorithms who are exploiting the network structure and not only past user behavior (`FBCA`, `LFBCA`, and `RWR`) are able to produce recommendations for every active user (Coverage = 1) whereas the rest produce recommendation only for a fraction of the active users. The most important point is that `FBCA` and `LFBCA` demonstrate the best results, with `LFBCA` dominating every other algorithm for almost all the snapshot under every metric.[4] These are the results of taking into account, additionally to past user preferences, the spatial distance of venues and the social interaction of

[4]Occasional exceptions happen at times when, according to Figure 7, the snapshots are extremely small, and thus there lacks data for building highly qualified recommenders.

the network. As time goes by and more information about past user preferences is known, traditional approaches are able to reduce the performance gap, but they are still inferior to `FBCA` and `LFBCA`. Finally, it is worth noticing that the algorithms in both datasets do not follow the same trend as the snapshots progress through time. The results for all algorithms in the Brightkite dataset are worse as time progresses, whereas for Gowalla, after an initial vibration in quality the results become better as the network becomes more mature. We believe that the explanation lies in Figure 7. In Brightkite the number of active users reaches a peak and it is relatively stable in days from 300-500, and then decreases. The reduction of the number of active users is part of the overall reduction of activity in the network (even active users, are less active than in other periods) and this has a direct impact on the number of recommendation that will be followed (or reported as followed) in the testing period. In Gowalla, we have very few users in the first months of its operation, so the very good behavior of the first months may be attributed to partly random effects, and as the number of active users grows, we have a stable increase in the quality of the recommendations.

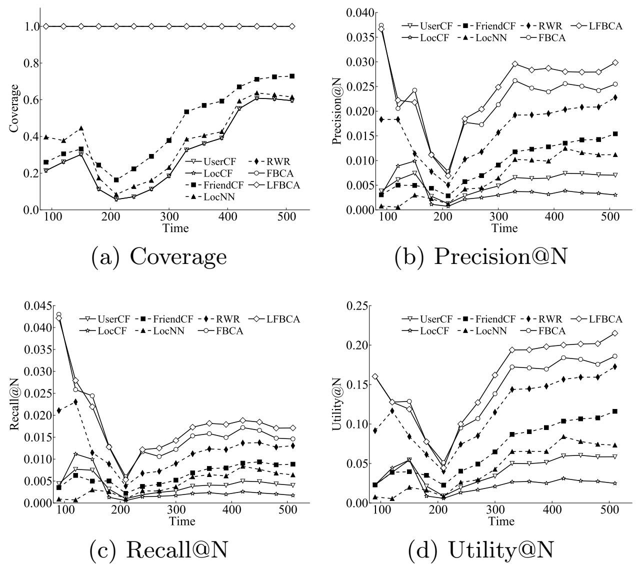


(a) Coverage (b) Precision@N

(c) Recall@N (d) Utility@N

Figure 8: Results for different snapshots of Brightkite

Figures 10 and 11 show how the performance of the algorithms changes as the number of recommendations, N, varies from 10 to 100 at typical snapshots ($t = 600$ for Brightkite and 270 for Gowalla). The Coverage remains stable for all algorithms (it decreases only slightly for `FriendCF`). Precision@N decreases as more recommendations are created, whereas Recall@N and Utility@N increase as the chance that a visited place has been recommended increases. The results are more or less the same for both datasets. Finally, in Figure 12 we present the impact of the distance filtering threshold d, that we use both in `FBCA` and `LFBCA`. In the interest of space we present results only for the Utility@N, but Precision@N and Recall@N follow the same trend. Figure 12(a) shows the effect of d in Brightkite and Figure 12(b) in Gowalla. As expected from the analysis of Section 2, a smaller d leads to better results, as users tend



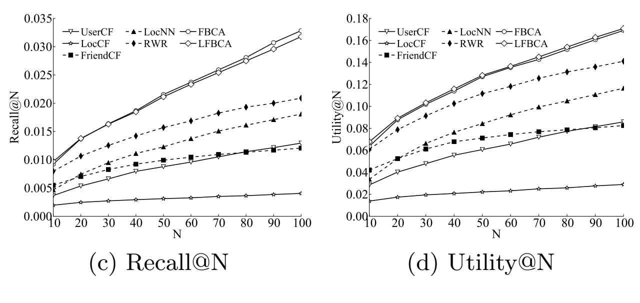
(a) Coverage (b) Precision@N

(c) Recall@N (d) Utility@N

Figure 9: Results for different snapshots of Gowalla

to visit venues in their vicinity. The effect of d for `LFBCA` in Brightkite is very limited, but due to the lack of a large number of active users, it could be the result of random factors.

In brief, the evaluation shows that the proposed algorithms `FBCA` and `LFBCA` manage to exploit the additional social and geographical information of the LBSNs and `LFBCA` outperforms the current state-of-the-art in all settings.



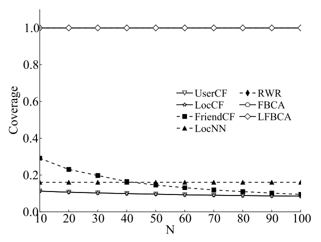
(a) Coverage (b) Precision@N

(c) Recall@N (d) Utility@N

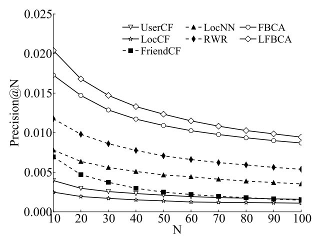
Figure 10: Results for Brightkite (varying N)

6. RELATED WORK

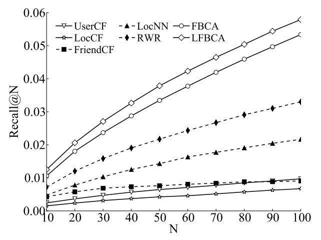
6.1 Recommender Systems based on User-Item Rating Matrix



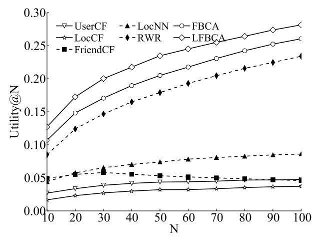
(a) Coverage



(b) Precision@N

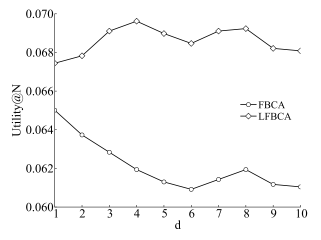


(c) Recall@N

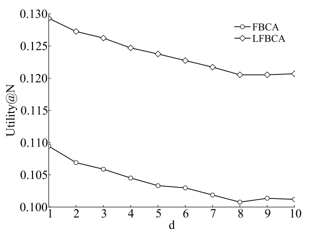


(d) Utility@N

Figure 11: Results for Gowalla for different values of N



(a) Utility@N in Brightkite



(b) Utility@N in Gowalla

Figure 12: The effect of d on utility for both datasets

Since the mid-1990s, recommender systems have become an important research area in computer science [1]. In the traditional setting, a recommender system deals with a user-item rating matrix. The matrix has one row for each user and one column for each item (the transpose matrix can be used depending on the algorithm). Each element of the matrix shows the preference of a user for an item. User u's rating on item i, r_{ui}, usually takes values from a small discrete domain (e.g., integers 0 to 5), modeling the degree of user's preference to the item (e.g., 5 means "love" and 0 means "hate"). Since it is very unlikely that all users have rated all items, the essential task of a recommender system is to predict the "missing" ratings, i.e., users' preference on their un-rated items. An item can be then recommended to a user if the estimated rating is high.

Collaborative filtering (CF) strategies offer personalized recommendations and they are very popular in practice [1, 13, 18, 20]. CF assumes that users with similar behavior in the past will like the same items (user-based CF) and that a user will prefer items that are similar to other items he has liked in the past (item-based CF). In user-based CF a rating r_{ui} is estimated by aggregating the ratings of other users $u_1, u_2, \cdots, u_n$ for item i, while the aggregation is based on the user-user similarity $\text{sim}(u, u_i)$. Analogously, *item-based CF* estimates r_{ui} based on u's past ratings on other items as well as item-item similarity.

Recently, model-based methods have been proposed for recommendation systems [7, 10, 21]. Suppose there are n users and m items, such methods assume that the actual rating matrix $R_{n\times m}$ can be well-approximated by a user profile matrix, $U_{n\times k}$, and an item profile matrix $V_{m\times k}$, i.e., $R_{n\times m} \approx U_{n\times k} \cdot V^T_{m\times k}$. U and V can be found via principle component analysis (PCA) and other low-rank techniques [21], or some iterative refinement process [7, 10].

Even though the above methods are quite popular in traditional settings, when adjusted for LBSNs (`UserCF` and `LocCF` in Section 3), they perform worse than the proposed algorithms, since they do not take into account social and spatial information.

6.2 Trust-based Recommender Systems

Traditional CF methods assume that users are independent, and they only take into account their past behavior. Trust-based methods aim to exploit the opinions that users have about other users. Briefly speaking, a *trust network* is a directed network showing users' mutual trust. Every directed edge $u \to v$ is attached a rating showing how much u trusts v. Such trust is designed to be transitive [5] and thus typically has a wider scope of application. In many cases the similarity cannot be computed (e.g., due to lack of information), however a trust value can still be inferred.

To predict a rating r_{ui}, TidalTrust [5] finds all the users who have rated item i using a breadth first search, and then aggregates all the ratings $r_{\cdot i}$ with direct or inferred trust values. MoleTrust [14, 15] further limits the breadth first search within a maximum depth. In a recent work, TrustWalker [6] proposed by Jamali and Ester combines user trust and item similarity into random walk processes. To predict the rating r_{ui}, TrustWalker performs a random walk in the trust network starting from user u. At any user node v, TrustWalker terminates reporting r_{vi} if v has rated i, otherwise TrustWalker takes a probabilistic action that either continues the random walk or terminates reporting some r_{vj} where item i and item j are similar; the probability is calculated based on user trust, item similarity, and the current random walk length. More recently, Yang et al. [22] studied *friend circles* to get more domain-specific trust values. In our location recommendation problem, we do not have any trust information between users, therefore the trust-based methods are not applicable.

6.3 Social Network-based Recommendation

Pham et al. [17] find user and item clusters in social networks and use such information to enhance CF methods. Random walk has also been exploited for recommendation. Yildirim and Krishnamoorthy [25] build a graph of items, in which each link is weighed by the similarity of its two owner items. Given a user u who has rated a set of items I, random walks on this item graph are adopted to find the items similar to I. In order to guarantee the connectivity of the item graph, two items are linked with a small weight even if the similarity is not computable. This approach makes the the method unsuitable for large datasets. Konstas et al. study CF methods on a music social network to predict music play-counts [9]. They create a heterogeneous graph containing users, music tracks, and tags, and show that a random walk on the graph outperforms traditional CF methods. Their technique is tailored for music data and it is not applicable in our problem.

6.4 Location-based Services and Recommenders

Scellato et al. study the problem of predicting friendship links using user check-in records [19]. They find that the effective link prediction on location-based services can greatly benefit from focusing only on the friends-of-friends and on the place-friends of a user.

Leung et al. propose a framework for location recommendation based on GPS trajectories [11]. A GPS trajectory is a sequence of time-stamped latitude/longtitude pairs, which are collected every a few seconds. *Stay ponits* (i.e., geographic regions at which a user spent sufficiently long time) are identified from the GPS log and treated as locations. Comparing to our user check-in data, GPS data are much more dense and contain more routine activities (e.g., travelling from home to office every morning). Based on such characteristics of GPS data, [11] performs a co-clustering on users and stay points to enhance CF recommendation.

Bao et al. study location recommendation with a location category hierarchy (e.g., food, shop, etc. and their subcategories) [2]. Their main observation is that different users may have different expertise on different types of locations, thus they must be treated differently in the recommendation. They develop a systematic approach that can (i) learn users' preferences/expertise from the category information, and (ii) recommend new locations within a user specified spatial range. These approaches rely heavily on the category hierarchy and are thus not applicable to our problem.

Levandoski et al. consider item recommendation using location information [12]. In their work an item may have spatial and non-spatial ratings, and may have a spatial attribute itself. A location hierarchy is then built to facilitate the recommendation process.

7. CONCLUSIONS

In this work we proposed two algorithms for recommending new venues to users in LBSNs. Unlike traditional approaches, the algorithms do not solely rely on past user preferences, but they also exploit the social relations of the network and the geographical location of the venues. The experimental evaluation shows that our approach outperforms traditional methods and the related state-of-the-art algorithms for recommendations in LBSNs.

Future work includes a deeper study of the network properties, the identification of communities and the incorporation of venue categories in the recommendation algorithm.

Acknowledgement

The authors would like to acknowledge the support from Hong Kong RGC (Grant No. HKU714212E) and EU/Greece funded COOPERATION: MoDiSSENSE Project. We would also like to thank the anonymous reviewers for their valuable comments.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[2] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *GIS*, 2012.

[3] P. Berkhin. Bookmark-coloring algorithm for personalized PageRank. *Internet Mathematics*, 3(1):41–62, 2006.

[4] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, 2011.

[5] J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland at College Park, 2005.

[6] M. Jamali and M. Ester. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In *KDD*, 2009.

[7] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, 2010.

[8] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, 2003.

[9] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR*, 2009.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, August:42–49, 2009.

[11] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. CLR: a collaborative location recommendation framework based on co-clustering. In *SIGIR*, 2012.

[12] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *ICDE*, 2012.

[13] G. Linden, B. Smith, and J. York. Item-to-item collaborative filtering. *IEEE Internet Computing*, Jan-Feb:76–80, 2003.

[14] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *CoopIS, DOA, ODBASE*, 2004.

[15] P. Massa and P. Avesani. Trust-aware recommender systems. In *RecSys*, 2007.

[16] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. A random walk around the city: new venue recommendation in location-based social networks. In *SocialCom*, 2012.

[17] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke. A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Sicence*, 17(4):583–604, 2011.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2010.

[19] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *KDD*, 2011.

[20] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering. *Advances in Artificial Intelligence*, 2009:19, 2009.

[21] M. G. Vozalis, A. Markos, and K. G. Margaritis. Collaborative filtering through SVD-based and hierarchical nonlinear PCA. In *ICANN*, 2010.

[22] X. Yang, H. Steck, and Y. Liu. Circle-based recommendation in online social networks. In *KDD*, 2012.

[23] M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *GIS*, 2010.

[24] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, 2011.

[25] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*, 2008.